【Requirement】Workflow Automation Backend Requirement

Parent Requirement

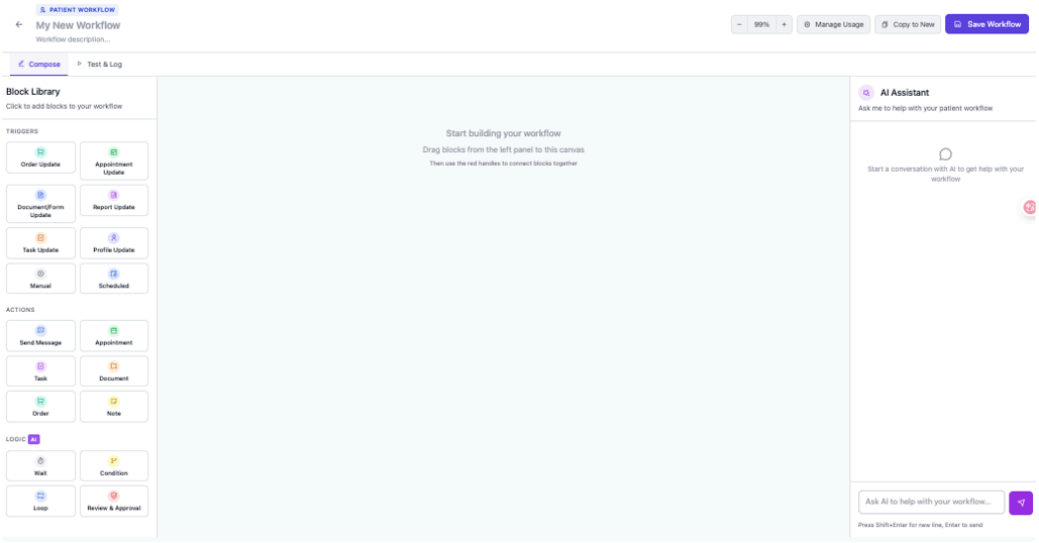
This is the back-end requirement for the workflow automation as described in 【Requirement】My Workbench | Workflow(Frent End)

Reference Code: workflow-composer.html

Summary

Overview

The Workflow Composer Backend enables healthcare professionals to create, execute, and manage automated patient care workflows through a visual interface. The backend system translates user-defined workflows into executable processes that can integrate with existing Vibrant EHR services (Orders, Calendar/Tasks, Patient Profiles, Messaging) to automate routine tasks and decision-making processes.



Key Capabilities

- **Natural Language to Workflow**: AI-powered workflow generation from user descriptions
- **Visual Workflow Execution**: Process workflows created through the drag-and-drop interface
- **Intelligent Logic Processing**: AI-driven decision making for conditions, loops, and reviews
- **Healthcare System Integration**: Seamless connection to existing internal systems
- **Real-time Workflow Orchestration**: Manage workflow state and execution flow

Primary Objectives

- Enable non-technical healthcare staff to automate complex care workflows
- Reduce manual repetitive tasks in patient care coordination
- Provide intelligent decision support through AI-powered logic evaluation
- Ensure reliable execution of time-sensitive healthcare processes
- Maintain audit trails for compliance and quality improvement

Success Criteria

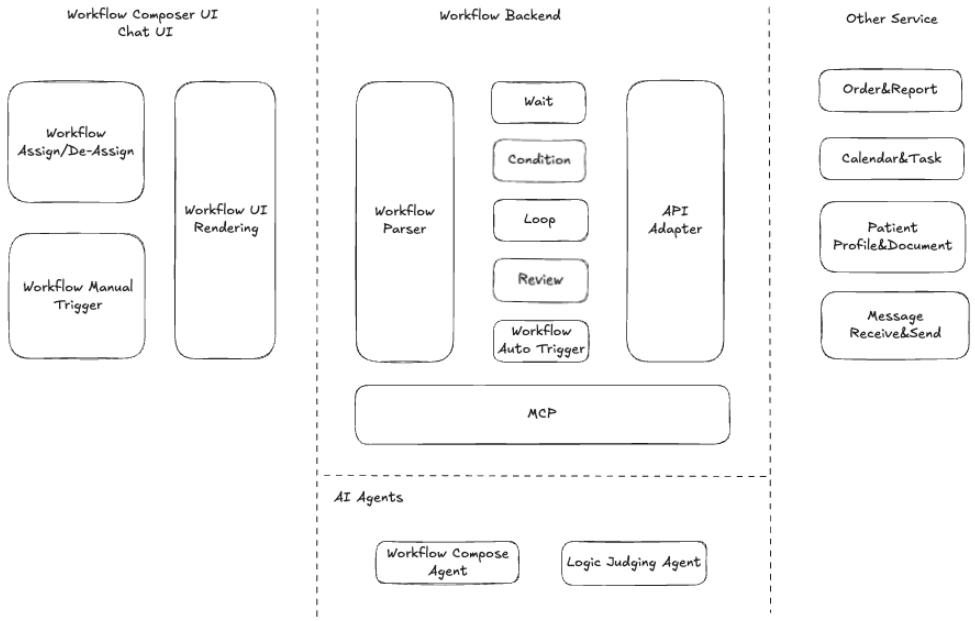
- Healthcare staff can create workflows using natural language descriptions
- Workflows execute reliably with proper error handling and recovery

- AI agents provide accurate logic evaluation for clinical decision points
- Integration with all existing internal systems (Orders, Calendar, Patient, Messaging)
- Comprehensive monitoring and logging for healthcare compliance requirements

Product Context

Basic Concepts

- **Frontend Implementation**: Complete visual workflow composer UI with drag-and-drop functionality
- **Supported Block Types**: 15+ block types including triggers (manual, scheduled, event-based), actions (messaging, appointments, orders), and logic blocks (refer to the prototype or front-end document for the supported blocks and status)
- **User Interface**: Canvas-based editor with workflow assignment/de-assignment capabilities, with AI copilot.
- **Integration Points**: Defined interfaces for connecting to Vibrant backend services.



User Needs

- **Workflow Creation**: Convert care protocols into automated workflows without technical expertise
- **Intelligent Assistance**: Get AI help to design optimal workflows from natural language descriptions
- **Reliable Execution**: Ensure workflows run consistently across different patient scenarios
- **Real-time Monitoring**: Track workflow progress and identify bottlenecks
- **Compliance Tracking**: Maintain detailed logs for healthcare regulatory requirements

User Stories and Use Cases

Core User Stories

US-001: AI-Assisted Workflow Creation

- **As a** healthcare administrator
- **I want to** describe a care process in natural language
- **So that** the system generates a workflow I can review, modify and deploy
- **Example**: "When a patient is discharged, wait 24 hours, then send a follow-up message asking about their recovery. If they report concerns, schedule a nurse call within 2 hours."

US-002: Visual Workflow Execution

- **As a** clinical staff member
- **I want to** manually trigger/assign workflows for specific patients
- **So that** automated care processes start when needed
- **Example**: Triggering a "New Patient Onboarding" workflow when I assign it to the new patient

US-003: Intelligent Decision Making

- **As a** workflow system
- **I need to** evaluate conditions and make decisions during execution
- **So that** workflows can branch based on patient data and responses
- **Example**: If patient questionnaire indicates pain level > 7, escalate to urgent care pathway

US-004: Scheduled Workflow Automation

- **As a** practice manager
- **I want** workflows to trigger automatically based on schedules or events
- **So that** routine processes happen without manual intervention
- **Example**: Weekly summary report for patient engagement rate

US-005: Workflow Monitoring and Control

- **As a** healthcare supervisor
- **I want to** monitor active workflows and intervene when needed
- **So that** I can ensure patient care quality and resolve issues
- **Example**: Pausing a workflow when a patient requests to opt out

Detailed Use Cases

UC-001: Natural Language Workflow Generation

1. User enters workflow description in natural language

2. Workflow Compose Agent analyzes the description using MCP context
3. Agent identifies required blocks, triggers, and logic flows
4. System generates workflow JSON with appropriate node types
5. User reviews and modifies the generated workflow on canvas
6. System validates workflow completeness and saves definition

UC-002: Event-Driven Workflow Execution

1. External system (Orders, Calendar, etc.) sends event notification
2. Workflow Auto-Trigger service evaluates if workflow should start
3. System creates new workflow execution instance
4. Workflow Parser processes the workflow definition
5. Execution engine begins processing nodes in sequence
6. System tracks execution state and progress

UC-003: Conditional Logic Evaluation

1. Workflow execution reaches a Condition node
2. Logic Judging Agent receives current workflow context via MCP
3. Agent evaluates condition against available data
4. Agent returns decision (true/false/custom logic)
5. Workflow execution continues down appropriate path
6. System logs decision rationale for audit purposes

UC-004: Vibrant Service Integration

1. Workflow execution reaches an action node (order, appointment, etc.)
2. API Adapter receives request with required parameters
3. The adapter translates request to appropriate Vibrant service API
4. Vibrant services processes request and returns status
5. The adapter normalizes the response and returns to the workflow engine
6. Workflow continues based on the success/failure status

Functional Requirements

Workflow Definition

A workflow is an automated sequence of functions that can be used in **chat quick actions** or data assembly, structured with **logic blocks**, and can be **triggered** manually, by schedule, or by events.

Workflow Types

- **Patient Workflows**: Applied to individual patients (may include future ones) for care coordination (e.g., post-discharge follow-up, order reminder, appointment reminders)
- **Practice Workflows**: Organization-wide processes for practice management (e.g., weekly patient engagement rate, monthly profit, etc.)

Key Components

There are 3 types of nodes for a workflow

- **Triggers**: Methods to initiate workflow execution (manual activation, scheduled execution, event-driven)
- **Actions**: Individual actions that perform specific tasks (send a message, create an appointment, place orders, etc.)
- **Logic**: Control flow elements that add intelligence (wait, condition, loop, review)

Here is a list of supported nodes for different types of workflows.

For details of how each block works, refer to the workflow execution section.

Block Type	Patient Workflows	Practice Workflows (To be finalized)	Description
TRIGGERS			
trigger-manual	✓	✓	Manual start (when the frontend tells the backend to execute)
trigger-scheduled	✓	✓	Scheduled execution (daily, weekly, monthly, also supports manual trigger)
trigger-order-events	✓	✗	Order status change events, supported status, same for

			the following triggers and actions
trigger-appointment-events	✓	✗	Appointment status change events
trigger-document-events	✓	✗	Document/form interaction events
trigger-report-events	✓	✗	Report status change events
trigger-task-events	✓	✗	Task status change events
trigger-profile-events	✓	✗	Patient profile change events
trigger-new-patient	✓	✗	New patient creation events
trigger-threshold	✗	✓	Threshold alerts (metrics exceed limits)
trigger-system-event	✗	✓	System-wide events
ACTIONS			
send-message	✓	✗	Send message/SMS/email to patient
appointment	✓	✗	Schedule appointment with patient
task	✓	✗	Assign a task to patient

document	✓	✗	Send document to patient
order	✓	✗	Place order
note	✓	✗	Add internal staff note
bulk-communication	✗	✓	Send messages to multiple patients
generate-report	✗	✓	Generate practice reports and analytics
send warning	✗	✓	Send a warning message to a team member
LOGIC BLOCKS			
wait	✓	✓	Wait for time or external input
condition	✓	✓	Conditional branching with AI evaluation
loop	✓	✓	Repeat operations with AI-driven logic
approval	✓	✓	AI-supported Human review and approval process

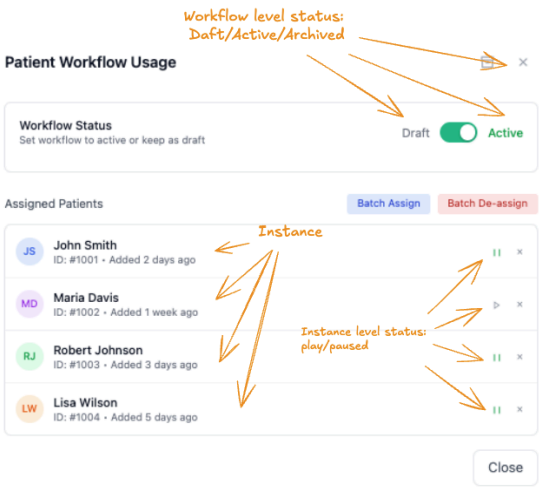
Core Workflow Properties

- **name**: Human-readable workflow title
- **description**: Detailed explanation of workflow purpose and process
- **type**: Classification as either "patient" or "practice" workflow

- **instance**: if there is executable(s) task pending for this workflow
  - A patient workflow can have **multiple live instances** at the same time (assigned to multiple patients)
  - A practice workflow can only have one live instance at the same time.
  - The backend should have a track log on all instances
  - The user will be able to test a draft workflow, so a dummy instance for testing also needs to be logged.

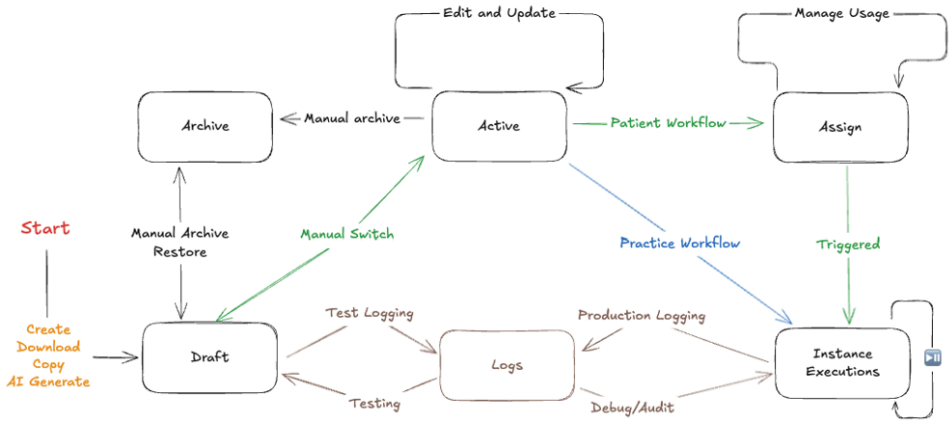
- **status**: Current workflow state
  - draft: Saved but not in use. Draft workflow can not have a live instance, all assigned instances will be paused, but the user can test a draft workflow with a dummy instance
  - active: Each activated workflow **instance** has a substatus
    - play: default active workflow status, will execute.
    - paused: A paused **instance** will not execute
  - archived: The workflow is marked as archived, archived workflow can not have live instance, and all instances will be removed.

Please refer to the front-end control panel demo to understand the status relationship:



- **owner**: User who created and is responsible for the workflow
- **version**: Current version number with change tracking
- **created_at/updated_at**: Timestamp tracking
- **assignment_rules**: Criteria for automatic patient/practice assignment(if automatecally assigned to all future patients)

Workflow Lifecycle



Creation Phase:

- **Start**: User begins workflow creation through one of 4 entry points:
  - Create: Build new workflow from scratch using visual composer
  - Download: Import existing workflow JSON file (i.e. downlaod from template library)
  - Copy: Duplicate existing workflow for modification
  - AI Generate: Use AI assistant to generate workflow from natural language description
- **Initial State**: All new workflows begin in "Draft" status
- System creates initial version with metadata and assigns unique workflow ID

Development Phase (Draft Status):

- **Testing**: Workflows can be tested in draft mode with test logging enabled, a draft workflow can be tested using a real patient workflow, but will not listen to actual events or clock, it also will not execute action blocks. The frontend shall provide dummy input for these blocks.
- **Edit and Update**: Iterative development with version tracking for each save
- **Validation**: System validates workflow structure, connections, and configurations
- **Test Logging**: Execution attempts logged separately for debugging and refinement

Activation Phase:

- **Manual Switch**: User manually promotes workflow from Draft to Active status, or switch back to Draft, if from Active to Draft, all instances are paused.
- **Activation Requirements**: The System validates workflow completeness before activation
- **Status Change**: Workflow becomes available for assignment and production use.

For Patient Workflows:

- **Assign**: Workflow assigned to specific patients, each assignment will create a live instance

- **Manage Usage**: Add/Remove assignment single/bulk, change instance status between pause and play
- **Triggered**: Workflow executions created based on trigger conditions
- **Instance Executions**: Individual workflow runs with unique execution IDs
- **Production Logging**: All executions logged for audit and monitoring

For Practice Workflows:

- **Direct Execution**: Practice workflows can be triggered directly without assignment, once activated, it automatically creates a live instance.
- **Triggered**: Automatic execution based on schedule or system events
- **Instance Executions**: Production workflow runs with full logging
- **Production Logging**: Comprehensive audit trail for compliance

Maintenance Phase:

- **Edit and Update**: Modifications create new versions while maintaining active status, affecting all live instances.
- **Debug/Audit**: Production logging analysis for optimization and troubleshooting
- **Performance Monitoring**: Execution metrics and success rate tracking

Retirement Phase:

- **Manual Archive**: User can manually archive workflows no longer needed, all instances are removed.
- **Archive State**: Workflow moved to Archive status, no longer available for new assignments
- **Manual Archive Restore**: Archived workflows can be restored to Draft status for reactivation
- **Historical Preservation**: All versions and execution logs maintained for compliance

Key State Transitions:

- Start → Draft (creation)
- Draft → Active (manual activation)
- Active → Archive (manual retirement)
- Archive → Draft (manual restoration)
- Any state → Logs (continuous logging throughout lifecycle)

Workflow Storage and Format

Primary Storage Format:

- Workflows stored as JSON documents containing complete workflow definition
- JSON structure includes: nodes, connections, parameters, metadata, and configuration

- **N8N format compatibility** is preferred but not mandatory

Storage Requirements:

- Complete workflow definition preservation
- Version history with diff tracking between versions
- Metadata indexing for search and filtering
- Backup and recovery capabilities
- Export functionality for workflow sharing

Workflow Validation:

- Structural validation (all nodes properly connected, no orphans)
- Business rule validation (required parameters completed)
- Integration endpoint availability checks
- Security and permissions validation

Workflow Orchestration and Execution

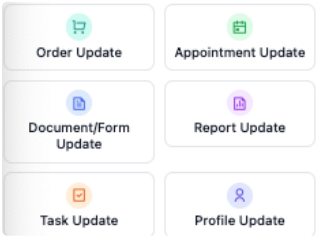
This section will cover the functions supported by each node by design and how they are connected together.

This section will introduce all 3 types of building blocks of a workflow, explain their connecting rule, and touch on the functions needed for running a workflow in the workflow engine part.

Trigger Blocks

Each workflow must have one and only one trigger to start with.

Event Triggers (Order, Appointment, Document, Report, Task, Profile)



- **Function**: Listen for specific system events and status changes for the **assigned patient**.
- **Parameters**: Event type checkboxes (e.g., Order Created, Appointment Accepted)
- **Validation**: At least one event type must be selected
- **Events Include**:

- Orders: Created, Modified, Payment, Lab Shipped, etc.

  Select Order Updates to Monitor:

  - [ ] Order Created
  - [ ] Order Modified
  - [ ] Order Redrawn
  - [ ] Payment Complete
  - [ ] Payment Failed
  - [ ] Payment Refund
  - [ ] Lab Shipped
  - [ ] Questionnaire Assigned
  - [ ] Missing Information
  - [ ] Test Not Performed
  - [ ] Order Canceled

- Appointments: Created, Accepted, Rejected, etc.

  Select Appointment Updates to Monitor:

  - [ ] Appointment Created
  - [ ] Appointment Accepted
  - [ ] Appointment Rejected
  - [ ] Appointment Tentative
  - [ ] Appointment Cancelled

- Documents: Viewed, Submitted

  Select Document/Form Updates to Monitor:

  - [ ] Document/Form Viewed
  - [ ] Form Submitted
  - [ ] Encounter Note Shared
  - [ ] Encounter Note Updated

- Reports: Available, Shared

  Select Report Updates to Monitor:

  - [ ] Report Available
  - [ ] Report Shared

- Tasks: Open, Completed

  Select Task Updates to Monitor:

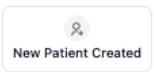
  - [ ] Task Open
  - [ ] Task Completed

- Profile: Updated

  Select Profile Updates to Monitor:

  - [ ] Patient Profile Updated
  - [ ] Internal Note Updated

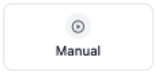
New Patient Trigger



This is a special type of event trigger, in terms of applying rule.

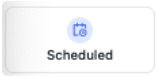
- **Function**: Auto-applies workflow to newly created patients
- **Parameters**: None
- **Validation**: None

Manual Trigger



- **Function**: Workflow started by user action (button click after assignment)
- **Parameters**: No configuration required
- **Validation**: None

Scheduled Trigger

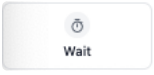


- **Function**: Time-based workflow execution
- **Parameters**:
  - Schedule type (X hours or X days)
  - Starting date and time
- **Validation**: Valid schedule configuration required

Logic Blocks

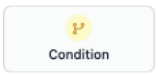
Embedded decision-making automation blocks, with optional AI judging feature.

Wait Block

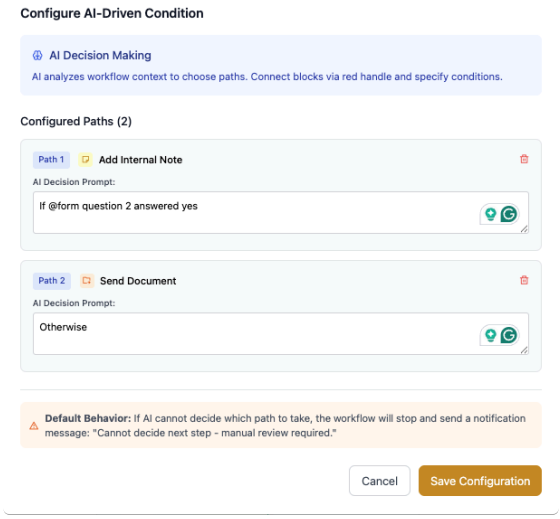


- **Function**: Pause workflow execution for a specified time and until all previous steps are finished. This block can accept max 5 previous steps.
- **Parameters**:
  - **Time mode**: `duration` (number ≥0), `unit` (minutes/hours/days)
- **Validation**:
  - Duration must be ≥0

Condition Block

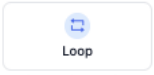


- **Function**: AI-driven conditional branching with multiple decision paths.
  This block can accept max 5 previous steps and have max 5 follow-up paths, and must include a default path.

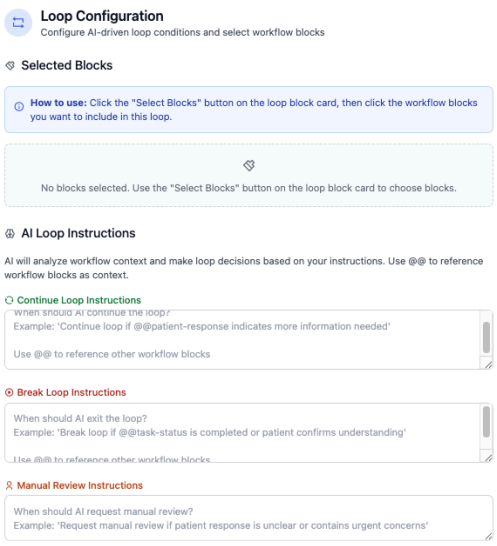


- **Parameters**:
  - **Paths**: Array of decision paths
  - Each path: `prompt` (AI instruction), `nextBlockId` (target block)
- **Logic**:
  - AI evaluates conditions using Logic Judging Agent
  - Fallback: "Cannot decide - manual review required"
  - See details on how to assemble prompt and call agent in next section
- **Validation**:
  - At least one path with non-empty prompt required
  - Each path must have valid prompt text
  - Maximum 5 paths per condition block

Loop Block

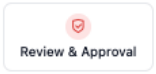


- **Function**: Optional AI-driven repetition with visual block selection

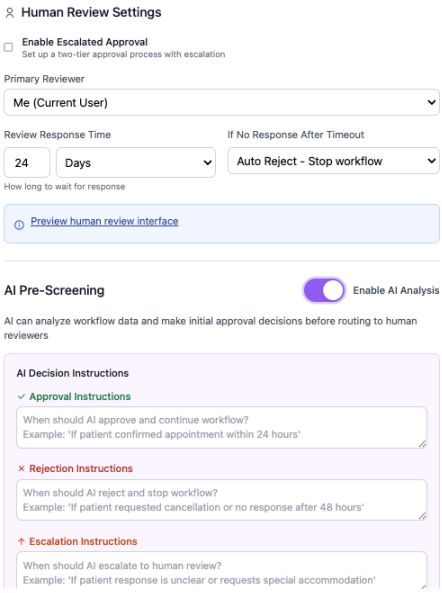


- **Parameters**:
  - **Count**: Number of iterations
  - **Enclosed Blocks**: Array of block IDs to repeat (selected via "rope" UI)
  - **(If AI enabled)**: prompt text to continue and break the loop and max loop time for AI
- **Logic**:
  - If AI is not enabled, loop given times for all enclosed blocks
  - If AI enabled, AI determines when to exit the loop based on context, unless it reaches max loop
  - See details on how to assemble prompt and call agent in next section
- **Validation**:
  - Count must be positive integer
  - At least one enclosed block required
  - Enclosed blocks must be valid and connected
  - If AI is enabled, continue and break instructions must have valid text, a max loop time must be set.

Review & Approval Block



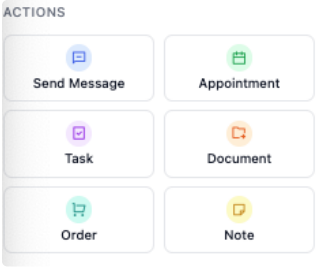
- **Function**: Human review process with optional AI pre-screening



- **Parameters**:
  - **AI Enabled**: Boolean for AI pre-screening
  - (If AI enabled): Prompts for decision making
  - **Primary Reviewer**: Required user selection
  - **Enable Escalation**: Boolean for two-tier approval
  - **Escalation Reviewer**: Secondary reviewer (if escalation enabled)
  - **Timeout Action**: auto-approve/auto-reject/escalate
  - **Timeout Duration**: Time before timeout action triggers
- **Logic**:
  - If approved, continue this execution; if rejected, stop this execution; if pending, keep waiting.
  - See details on how to assemble prompt and call agent in next section

- **Validation**:
  - Primary reviewer required
  - If escalation enabled, escalation reviewer required
  - Timeout duration must be positive
  - Timeout action selection required

Business Action Blocks



The following actions reuse the same functions as defined in Chat Quick Action. Assuming no new backend service needs to be developed, the target patient will be fetched when the workflow is assigned.

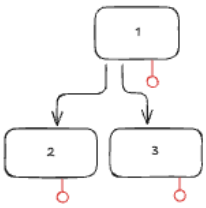
- **Make Appointment** - *[Schedule appointments with providers]*
- **Assign Task** - *[Create tasks for team members]*
- **Send Document**- *[Share documents with patients]*
- **Place Order**- *[Create lab/medication orders]*
- **Add Note** - *[Internal staff notes]*
- **Send Message -** *[Send Chat message]*

[Placeholder for practice workflow blocks]

Block Connection

This explain how blocks are connected togehter and their logic relationship and limiations.

This is a dummy workflow connected together to explain the possible relationship of blocks, in this dummy workflow:



Terms:

- Block1 is the **previous** of Block2 and Block3
- Both Block2 and Block3 are the **follow-up** of Block1
- Block1⟶Block2 is a **sequential** step (Defined as a follow-up to a previous)
- Block1⟶Block3 is a sequential step

- Block2 and Block3 are **parallel** step (Defined as multiple follow-ups to a same previous)
- The red handle means addtional follow-ups can be added to this block

Connection rules for trigger and action blocks

Sequential Connections

- **Trigger blocks**: Cannot have previous blocks (workflow start points only)
- **Action blocks**: Can have only 1 previous block.
- **Execution blocking**: Follow-up blocks wait until previous block reaches "finished" status
- **User-defined “Finished”**: Users define what constitutes "finished" for each block
- **Status selection**: When creating connections, users select which status(s) of the previous block indicate “Finished”
- **Available statuses**: Same status options as corresponding trigger events for that block type; multi-selection is allowed.

Multiple Output Connections

- **Up to 5**: Trigger and Action blocks can have up to 5 parallel follow-up blocks.
- **Status-based routing**: Different follow-up blocks can trigger on different “Finished“ statuses
- **Non-exclusive configuration**: Multiple follow-ups can use overlapping statuses
- **Example**: Block1 → StatusA triggers Block2, Block1 → StatusB triggers Block3
- **Simultaneous execution**: If Block1 reaches both StatusA and StatusB, both Block2 and Block3 execute

Parallel Connections

- **Non-interlocked execution**: Parallel blocks can execute simultaneously
- **Independent processing**: No waiting dependencies between parallel branches
- **Shared trigger**: Multiple blocks can start from same completion status

Connection Configuration

- **Per-connection settings**: Each output connection has independent status trigger configuration
- **Flexible routing**: Single block can have max of 5 different status-based outputs
- **No mutual exclusivity**: Output connections can overlap in their trigger conditions

Connection rules for logic blocks

Condition Blocks

- **Multiple inputs**: Support up to 5 input connections
- **Multiple outputs**: Up to 5 AI-driven decision paths per condition block

- **Path configuration**: Each output connection requires AI prompt configuration
- **Finished status**: Determined by AI evaluation completing and selecting output path

Loop Blocks

- **No direct connections**: User will select which block(s) to enclose in a loop
- **Enclosed block management**: Selected blocks stored in `enclosedBlocks` array
- **Internal execution**: Enclosed blocks execute as group for each loop iteration
- **Finished status**: User can include a prompt for AI to determine when to continue, break the loop, or report to a human. Loop completes when exit conditions are met or max iterations reached, determined by AI or user configuration of max loop count.

Wait Blocks

- **Multiple previous**: The Wait block can have up to 5 previous blocks.
- **2-Stage Waiting**: When connected to multiple previous blocks, it will wait until all previous blocks are “finished“ before starting the in-block countdown.
- **Time-based completion**: The wait block Finished when the wait duration expires
- **Multiple follow-ups**: The wait block can have up to 5 parallel follow-up blocks, like trigger and action blocks, but they all share the same Finished status.

Review & Approval Blocks

- **Single previous**: Review & Approval block can have only 1 previous block, same as action blocks
- **Multiple follow-ups**: Review & Approval block can have up to 5 parallel follow-up blocks, like trigger and action blocks, but they all share the same Finished status.
- **Finished status**: Completed when an approval decision is made by human/AI, or timeout action.

Workflow Engine (Just for reference)

Preferred function list for workflow execution, as a reference for development technical design.

This section is just for reference

Flowchart

Workflow Engine Architecture

Workflow Parsing and Preparation

JSON Definition Processing:

- Accept N8N-compatible (the compatibility is not mandatory) workflow JSON from the frontend
- Extract workflow metadata (name, type, version, owner, triggers)
- Parse node definitions and validate all required parameters are present
- Build execution graph showing node dependencies and flow paths
- Identify parallel execution branches and merge points
- Validate workflow completeness (no orphaned nodes, proper connections)
- Cache parsed workflow for execution optimization

Pre-execution Validation:

- Verify all external integrations are available and authenticated
- Check patient/practice assignments are valid and active
- Validate trigger conditions are properly configured
- Ensure all required approvers/reviewers are available for approval nodes
- Confirm AI agents (Workflow Compose, Logic Judging) are accessible via MCP
- Generate execution plan with estimated duration and resource requirements

Execution Lifecycle Management

Workflow Initiation:

- **Start**: Create unique execution instance with execution ID
- **Context Initialization**: Load patient data, practice data, previous execution history
- **State Setup**: Initialize execution state tracking for all nodes
- **Logging**: Begin comprehensive audit trail with start timestamp and trigger details
- **Notification**: Inform relevant users that workflow has started

Active Execution Control:

- **Pause**: Immediately stop processing at current node, preserve all state data
  - User can pause manually from monitoring interface
  - System can auto-pause on errors or approval requirements
  - Save exact pause point for seamless resumption
- **Resume**: Continue from exact pause point with preserved context
  - Validate system state hasn't changed critically since pause
  - Re-authenticate integrations if needed
  - Continue with same execution ID and audit trail
- **Stop**: Gracefully terminate execution with cleanup
  - Complete current node processing before stopping

    - Save final state and mark execution as "cancelled"
    - Notify all stakeholders of cancellation
- **Emergency Abort**: Immediate termination for critical issues
    - Stop all processing immediately without cleanup
    - Mark execution as "aborted" with reason
    - Trigger alerts to system administrators

Completion Handling:

- **Success**: Mark execution as completed, save final state, trigger completion notifications
- **Failure**: Capture error details, save partial results, trigger failure notifications
- **Timeout**: Handle workflows exceeding maximum execution time limits

Node Execution Sequencing

Sequential Processing:

- Execute nodes in order defined by workflow connections
- Wait for each node to complete before proceeding to next
- Pass execution context and data between nodes
- Handle node-specific timeouts and retry logic
- Support conditional branching based on node outputs

Parallel Execution Paths:

- **Branch Creation**: When workflow splits into multiple paths, create separate execution threads
- **Independent Processing**: Each branch executes independently with shared context
- **Resource Management**: Track resource usage across all parallel branches
- **Branch Monitoring**: Monitor progress of each parallel branch separately
- **Merge Handling**: When branches converge, wait for all branches to complete before proceeding
- **Failure Propagation**: If one branch fails, determine impact on other branches and overall workflow

Node-Specific Processing Requirements:

Wait Nodes:

- **Time-based**: Schedule workflow continuation at specified future time
- **Input-based**: Pause execution until required external input received
- **Event-based**: Wait for specific system events or webhook notifications
- **Timeout handling**: Resume execution if wait conditions not met within limits

Condition Nodes:

- **Data Gathering**: Collect all relevant context data for AI evaluation
- **AI Integration**: Send context to Logic Judging Agent via MCP
- **Decision Processing**: Receive AI decision and rationale
- **Path Selection**: Route execution to appropriate next node based on decision
- **Fallback Handling**: Handle cases where AI cannot make decision

Loop Nodes:

- **Iteration Tracking**: Maintain loop counter and iteration history
- **Enclosed Block Execution**: Execute all enclosed blocks for each iteration
- **Exit Condition Evaluation**: Check loop exit conditions after each iteration
- **Context Management**: Maintain separate context for each iteration while preserving overall workflow context
- **Infinite Loop Protection**: Enforce maximum iteration limits

Approval Nodes:

- **Review Queue Creation**: Add workflow to appropriate reviewer's queue
- **Notification Dispatch**: Notify assigned reviewers of pending approval
- **AI Pre-screening**: If enabled, get AI recommendation before human review
- **Decision Capture**: Record approval/rejection decision with rationale
- **Escalation Processing**: Handle escalation to secondary reviewers if configured
- **Timeout Management**: Apply timeout actions if approval not received within time limits

Action Nodes:

- **API Integration**: Call appropriate external system APIs via API Adapter
- **Parameter Preparation**: Format workflow context data for external system consumption
- **Execution Monitoring**: Track API call success/failure and response data
- **Result Processing**: Incorporate API responses back into workflow context
- **Error Handling**: Retry failed API calls and handle permanent failures

State and Context Management

Execution State Tracking:

- **Workflow Level**: Overall status, current node, execution start time, elapsed time
- **Node Level**: Individual node status, execution time, input/output data, error details
- **Branch Level**: Status of parallel execution branches and their relationships
- **Data Context**: All workflow variables and data collected throughout execution

Context Data Management:

- **Patient Context**: Patient ID, demographics, medical history, current medications, recent interactions
- **Practice Context**: Organization settings, team members, operational parameters
- **Workflow Context**: Data passed between nodes, user inputs, external system responses
- **Historical Context**: Previous workflow executions for same patient/process
- **Session Context**: User who initiated workflow, timestamps, system state

Data Persistence Requirements:

- **Real-time Updates**: Save state changes immediately as they occur
- **Recovery Data**: Maintain sufficient data to resume from any point
- **Audit Trail**: Complete history of all state changes with timestamps and reasons
- **Data Retention**: Archive execution data according to compliance requirements

Error Recovery and Rollback

Error Detection and Classification:

- **Node Failures**: Individual node execution errors with specific error codes
- **Integration Failures**: External system unavailability or API errors
- **Data Issues**: Invalid data formats, missing required information
- **Timeout Errors**: Operations exceeding configured time limits
- **System Errors**: Infrastructure or platform-level failures

Recovery Strategies:

- **Automatic Retry**: Retry failed nodes with exponential backoff
- **Manual Intervention**: Pause workflow for manual error resolution
- **Skip and Continue**: Mark failed node as skipped and continue execution
- **Alternative Path**: Route to alternative workflow path if available
- **Rollback to Checkpoint**: Return to last successful state before error

Rollback Capabilities:

- **Node-level Rollback**: Undo actions of specific failed node
- **Checkpoint Rollback**: Return to predefined safe state in workflow
- **Full Rollback**: Completely undo all workflow actions and return to initial state
- **Partial Rollback**: Undo specific actions while preserving others
- **Compensating Actions**: Execute reverse operations to undo completed actions

Recovery Decision Matrix:

- **Critical Errors**: Immediate workflow termination with full rollback

- **Recoverable Errors**: Pause for manual intervention or automatic retry
- **Warning Conditions**: Continue with logging but flag for review
- **Data Inconsistencies**: Attempt data correction or request manual validation

---

Global Validation Rules

Workflow Structure Validation

1. **Single Trigger Rule**: Exactly one trigger block per workflow (enforced with amber warning)
2. **Orphan Detection**: All blocks must be reachable from trigger

Block Configuration Validation

1. **Required Fields**: Each block type has mandatory parameters
2. **Data Types**: Numeric fields validated for positive integers
3. **Text Fields**: Non-empty validation for required text inputs
4. **Selection Fields**: Valid options from predefined lists
5. **Configuration Status**: Blocks marked as configured/unconfigured

Runtime Validation

1. **Parameter Completeness**: All required parameters configured before activation
2. **Reference Integrity**: Block references (nextBlockId) point to valid blocks
3. **Loop Safety**: Loop blocks have exit conditions to prevent infinite loops
4. **Path Completeness**: Condition blocks have valid paths for all scenarios

///

[Working In Progress]AI Agent Integration

Workflow Compose Agent

- Accept natural language workflow descriptions
- Access MCP context for available blocks and capabilities
- Generate structured workflow JSON definitions
- Provide workflow optimization suggestions
- Support iterative refinement through conversation

Logic Judging Agent

- Receive workflow execution context via MCP

- Evaluate complex conditions and business rules
- Make decisions for the Condition, Loop, and Review nodes
  - [How to assemble the prompt and call the MCP]
- Provide explanation and rationale for decisions
- Handle uncertainty and edge cases gracefully

MCP (Model Context Protocol) Framework

- Provide context about available workflow blocks and their functions
- Supply real-time workflow execution data to AI agents
- Manage data flow between agents and workflow engine
- Ensure data privacy and security in agent communications
- Support agent capability discovery and registration

Not Nontrivial Stuff

Visibility

Execution Tracking

- Real-time workflow execution status and progress
- Node-level execution tracking with timestamps
- Workflow completion status and outcome logging
- Error detection and failure point identification
- Execution duration and performance metrics

Audit and Compliance

- Comprehensive workflow execution logs
- Decision audit trails for AI-powered logic nodes
- User action tracking and approval chains
- Data access and modification logging

Workflow Analytics

- Workflow usage patterns and frequency analysis
- Success/failure rate tracking
- Performance bottleneck identification

Integration

API Calling Capabilities

- Standardized interface for internal system connections
- Support for Order & Report system integration
- Calendar & Task system connectivity
- Patient Profile & Document system access
- Message Receive & Send system integration

External System Communication (future plans)

- RESTful API calls to integrated systems
- Example: get wearable data.

Permissions

Access Control

- Practice admin only, referring to role design here

Workflow Collaboration

- Multi-user workflow editing - out of scope

Non-Functional Requirements

[Security, reliability, compliance requirements without specific metrics]

Data Requirements

[What data needs to be stored and managed, without specifying storage solutions]

Constraints and Assumptions

[Business constraints, regulatory requirements, assumptions]

Acceptance Criteria

[How we'll know the feature is successful from a